AMENDMENTS TO THE SPECIFICATION

In the Specification:

(1) On page 1, please amend the Title as follows:

Title: ~~Method Implementation~~Methods and Systems for Expressing and Interpreting Associations Between Declarations and Implementations in a Language Neutral Fashion

(2) On page 1 at line 9, please amend the specification as follows:

Software programs are commonly written in a high-level programming language, such as ~~visual basic~~VISUAL BASIC, C++, COBOL, Pascal, Smalltalk, or the like. The high-level language statements or instructions of the program (*e.g.*, source code) are then translated or compiled into coded instructions (*e.g.*, native or object code), which are executable by the computer. Typically, a software program known as a compiler is used for this translation. Processes for compiling source code into executable object or native code are well known in the art. The compiler initially performs lexical analysis on the source code to separate the source code into various lexical structures of the programming language (generally known as tokens), such as keywords, identifiers, operator symbols, punctuation, and the like. Syntactical analysis is then performed in which the compiler groups the tokens into various syntax structures of the programming language, such as expressions, declaration statements, loop statements, procedure calls, and the like. Thereafter, the compiler generates and optimizes code for each of these structures.

(3) On page 6 beginning at line 26, please amend the specification as follows:

~~Fig. 10 is a schematic diagram further illustrating the exemplary expression of an association between a declaration and an implementation of Fig. 9, as well as an interpretation thereof in accordance with the invention;~~

(4) On page 6 beginning at line 29, please amend the specification as follows:

Fig. [[11]]10 is a schematic diagram illustrating an exemplary source compiler and an exemplary runtime system in accordance with another aspect of the invention; and

(5) On page 7 beginning at line 1, please amend the specification as follows:

Fig. [[12]]11 is a schematic diagram illustrating an exemplary operating environment in which one or more aspects of the invention may be implemented.

(6) On page 7 at line 20, please amend the specification as follows:

Referring now to the drawings, Fig. 1 illustrates an[[d]] exemplary method 2 of expressing an association between a declaration and an implementation according to one aspect of the invention. While the exemplary method 2 is illustrated and described herein as a series of acts or events, it will be appreciated that the present invention is not limited by the illustrated ordering of acts, as some acts may occur in different orders and/or concurrently with other acts apart from that shown and described herein, in accordance with the invention. In addition, not all illustrated acts may be required to implement a methodology in accordance with the present invention. The method 2 may find utility in association with source compilers and other systems illustrated and described in greater detail herein. However, it will be appreciated that the exemplary method 2, as well as

other methodologies according to the invention, may be implemented in association with the apparatus and systems illustrated and described herein as well as in association with other systems not illustrated.

(7) On page 16 at lines 3 and 8, please amend the specification as follows:

As further illustration of the invention, Fig[[s]]. 9 ~~and 10~~ illustrates an example of the association expression and interpretation aspects of the invention. The association between a declaration and an implementation may be expressed in an explicit, language neutral fashion via a system in a source compiler (not shown) as described above. The source compiler may emit an IL code representation of a source language program as well as a metadata component 300, a portion of which is illustrated in Fig[[s]]. 9 ~~and 10~~. The metadata 300 may include various information which may be organized into one or more tables, such as a class table 302, a virtual method table 304, and an override association table 306.

(8) On page 16 at line 29, please amend the specification as follows:

Referring now to Figs. 4, 5, and 9, the entry 330 in the override association table 306 may be used to express the association 130, whereby a runtime system may interpret the association of the declaration 110 for a method named "M" in the interface I1 100 with the appropriate implementation code body 120 for the virtual method M 122 of the inheriting class C 104. The override association table 306 and the entry 330 thus provide a language neutral explicit association in a situation in which a runtime system employing only name or signature matching may have difficulty in establishing the proper or desired association at runtime. Referring also to Fig. [[10]]9, the entry 332 in the override association table 306 may be used to express the association 132, whereby the runtime system may interpret the association of the declaration 112 in the interface I2

102 for a method named "M" with the appropriate implementation code body 124 for the virtual method M 126 of the inheriting class C 104.

(9) On page 17 at lines 4 and 6, please amend the specification as follows:

An exemplary runtime system 400 is illustrated in Fig. [[11]]10 for running a software program. The program may comprise a source code component 420 created or written in a source code language (*e.g.*, ~~visual basic~~ VISUAL BASIC, C++, C#, java script, APL, COBOL, Pascal, Eiffel, Haskell, ML, Oberon, Perl, Python, Scheme, Smalltalk, Objective Camel, and the like), an intermediate language (IL) component 422 (*e.g.*, MSIL or the like), and a native code component 424, wherein the native code component 424 comprises instructions which may be executed or directly operated on by a processor 430 in a computer system 402.

(10) On page 17 at line 26, please amend the specification as follows:

The IL component 422 comprises intermediate language instructions representative of functions, methods, variables, etc. associated with the software program, and the metadata 428 may include descriptions of types, classes, references to external assemblies, record version information such as author, and the like, which may be provided in the form of one or more tables (*e.g.*, Fig[[s]]. 9 ~~and 10~~). The assembly 426 may be presented to the runtime system 400 by the source compiler 418 as a unit of deployment for execution therein in the form of a file, such as a .exe or a .dll file, which comprises the IL component 422 and the metadata component 428. The runtime system 400 may load the assembly 426 or portions thereof into memory for JIT compilation and execution via a class loader component 434, which may load classes or types within the assembly 426 on a class-by-class basis using a layout engine 454, wherein the associated IL code for a class (*e.g.*, from the IL component 422) and the associated metadata for the

class or type (*e.g.*, from the metadata component 428) are loaded into memory as the class or type is needed.

(11) On page 19 at line 26, please amend the specification as follows:

In order to provide a context for the various aspects of the invention, Fig. [[29]]11 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the various aspects of the present invention may be implemented. While the invention has been described above in the general context of software tools and computer-executable instructions of a computer program that runs on a computer and/or computers, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks and/or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods may be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, as well as personal computers, hand-held computing devices, microprocessor-based or programmable consumer electronics, and the like. The illustrated aspects of the invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. However, some, if not all aspects of the invention can be practiced on stand-alone computers. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

(12) On page 20 at line 15, please amend the specification as follows:

With reference to Fig. [[29]]11, an exemplary environment [[ ]]for implementing various aspects of the invention includes a conventional personal or server computer 520,

including a processing unit 521, a system memory 522, and a system bus 523 that couples various system components including the system memory to the processing unit 521. The processing unit 521 may be any of various commercially available processors. Dual microprocessors and other multi-processor architectures also can be used as the processing unit 521. The system bus 523 may be any of several types of bus structure including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of conventional bus architectures. The computer memory may include read only memory (ROM) 524 and random access memory (RAM) 525. A basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within the computer 520, such as during start-up, is stored in ROM 524.

(13) On page 35, please amend the Abstract of the Invention as follows:

Abstract Of The Invention

Methods and systems are ~~disclosed~~provided for expressing one or more associations between source language declarations and implementations in a language neutral fashion. A determination is made as to whether a source language association rule related to a declaration is different from a default association rule for a target runtime. If so, an override association is expressed between the declaration and the implementation, and if not, a default association is expressed. Methods and systems are ~~further disclosed~~also provided for interpreting an association between ~~and~~ a declaration and an implementation in a runtime system, wherein a determination is made as to whether the association comprises an override association. If so, the association is interpreted according to an override association rule for the runtime system, and if not, the association is interpreted according to a default association rule.

AMENDMENTS TO THE DRAWINGS

In the Drawings:

(1) In Figure 2, the symbol ">1" is replaced with the words "MORE THAN ONE".

(2) Figure 10 has been deleted.

(3) Figures 11 and 12 have been renumbered as Figures 10 and 11, respectively.